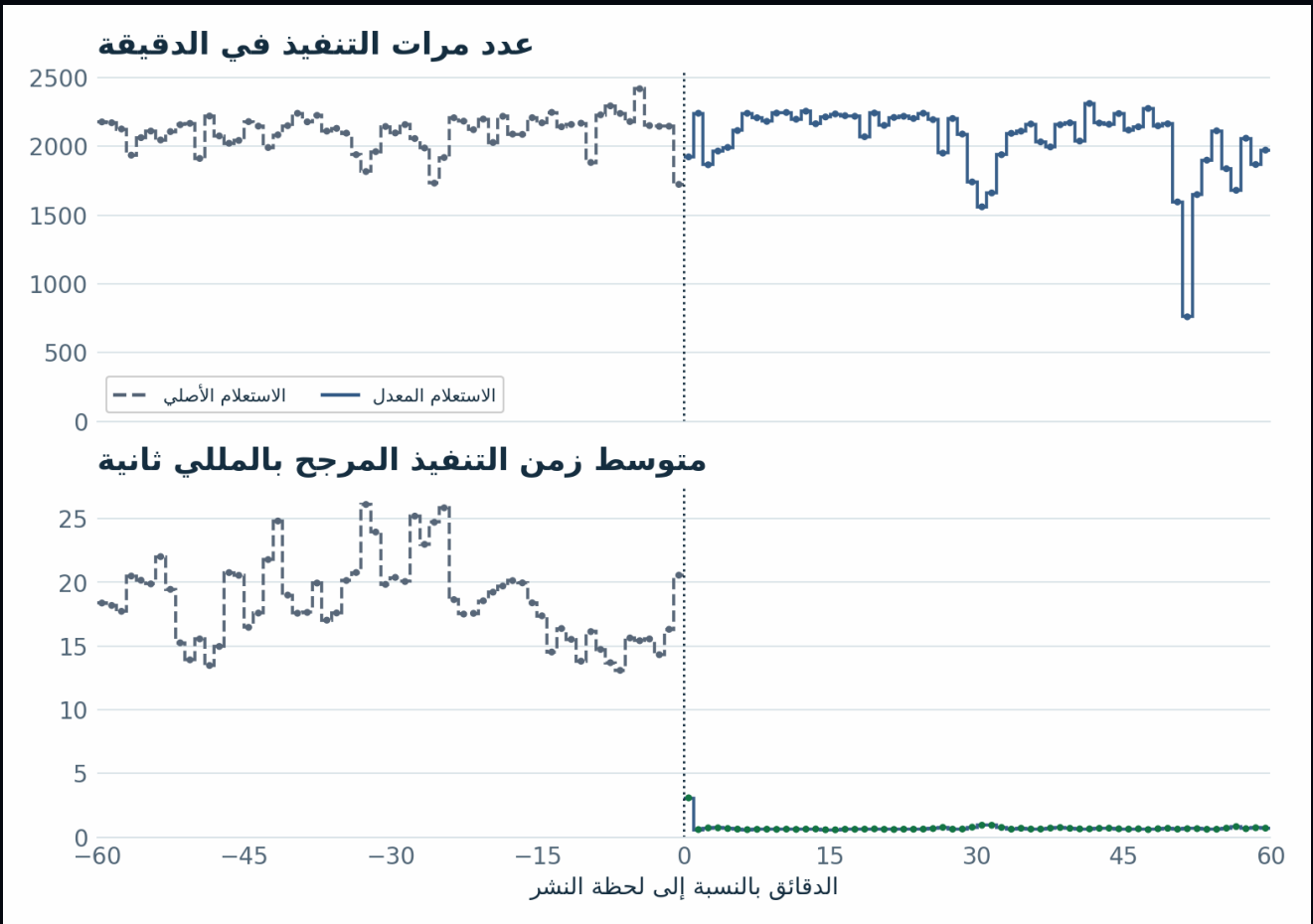
لماذا انخفض زمن استعلام MariaDB من 18.43 إلى 0.70 مللي ثانية مع أن السم كان يطابق صفاً واحداً؟

Aurélien LEQUOY · 21 نوفمبر 2026

PMACONTROL MARIADB PERFORMANCE SQL BENCHMARK OBSERVABILITY



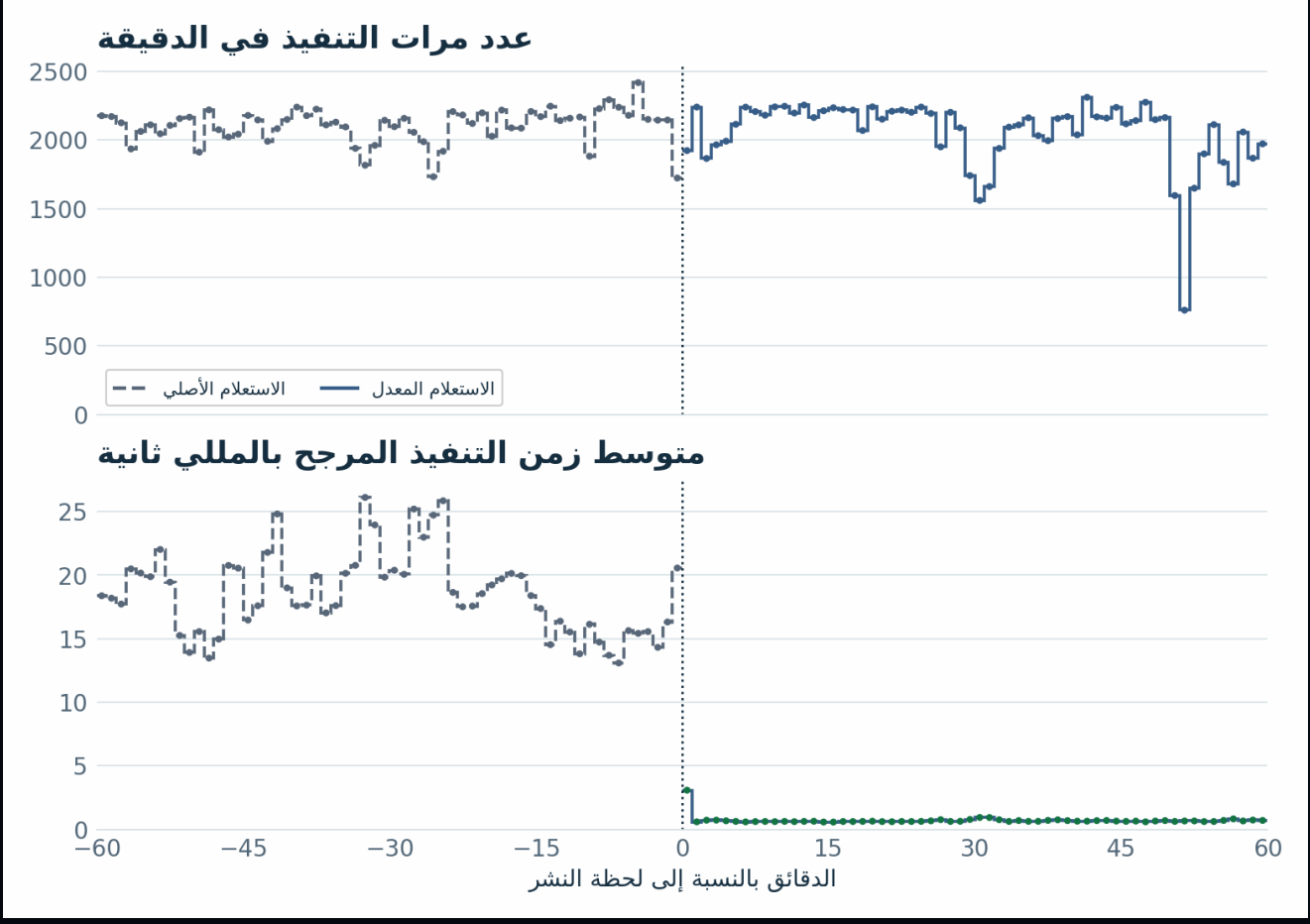
تضم `ts_variable` **3,759 اسماً مميزاً بين 3,791 صفاً**، لذا كانت انتقائية العمود `name` عالية أصلاً. بالنسبة إلى `mysql_available` كان السم يطابق صفاً واحداً بالضبط. وأبقى الشرط `v.`from` = 'mysql_server'` على الصف نفسه. كان من الطبيعي توقع تحسن ضئيل. لكن عند نحو 2,100 استدعاء في الدقيقة **انخفض متوسط زمن الاستجابة في الإنتاج من 18.43 إلى 0.70 مللي ثانية**. فماذا كسبت MariaDB مع أن الشرط الثاني لم يستبعد أي صف إضافي لهذا المقياس؟

قبل تسجيل توافر خدمة، يقرأ PmaControl آخر قيمة لمعرفة ما إذا تغيرت حالتها. استغرقت هذه القراءات في الساعة السابقة للنشر **2326 ثانية من زمن SQL التراكمي**. وفي الساعة

التالية استغرقت **87 ثانية** مع عدد متقارب من الاستدعاءات. التغيير الوحيد كان شرطاً في الربط.

ساعة قبل النشر وأخرى بعده

بدأ المجمّع استخدام الاستعلام المعدل يوم 21 نوفمبر 2026 عند **17:26:47 بتوقيت باريس**. وقد أدى تغير نص SQL إلى ظهور بصمة جديدة في Performance Schema. وعند تتبع البصمة القديمة وحدها يبدو أن عدد مرات التنفيذ هبط إلى الصفر. لذلك جمعنا فروق عدادات البصمتين لكل دقيقة لتتبع عمليات القراءة الفعلية.



تتكون كل سلسلة من 120 نقطة، نقطة لكل دقيقة. يعرض الرسم العلوي عدد مرات التنفيذ في كل دقيقة. ويبين الرسم السفلي متوسط زمن الاستدعاءات نفسها. يمثل الخط الرأسي المنقط وقت النشر. يشمل الحساب استدعاءات النسخة الأخرى حول لحظة الانتقال.

في الجدول، يشير الأخضر (✓) إلى التحسن والأحمر (✗) إلى التراجع عندما تتجاوز القيمة الكبرى ضعف الصغرى. ويشترط للدد أيضاً فرق لا يقل عن مللي ثانية واحدة. يصف عدد الاستدعاءات حجم العمل. تساعد هذه العتبة على قراءة النتائج. تميّز الخطوط في الرسوم بين الفترتين. تُقارن كل نقطة زمن بعد النشر بالمتوسط المرجح للساعة السابقة. يستخدم

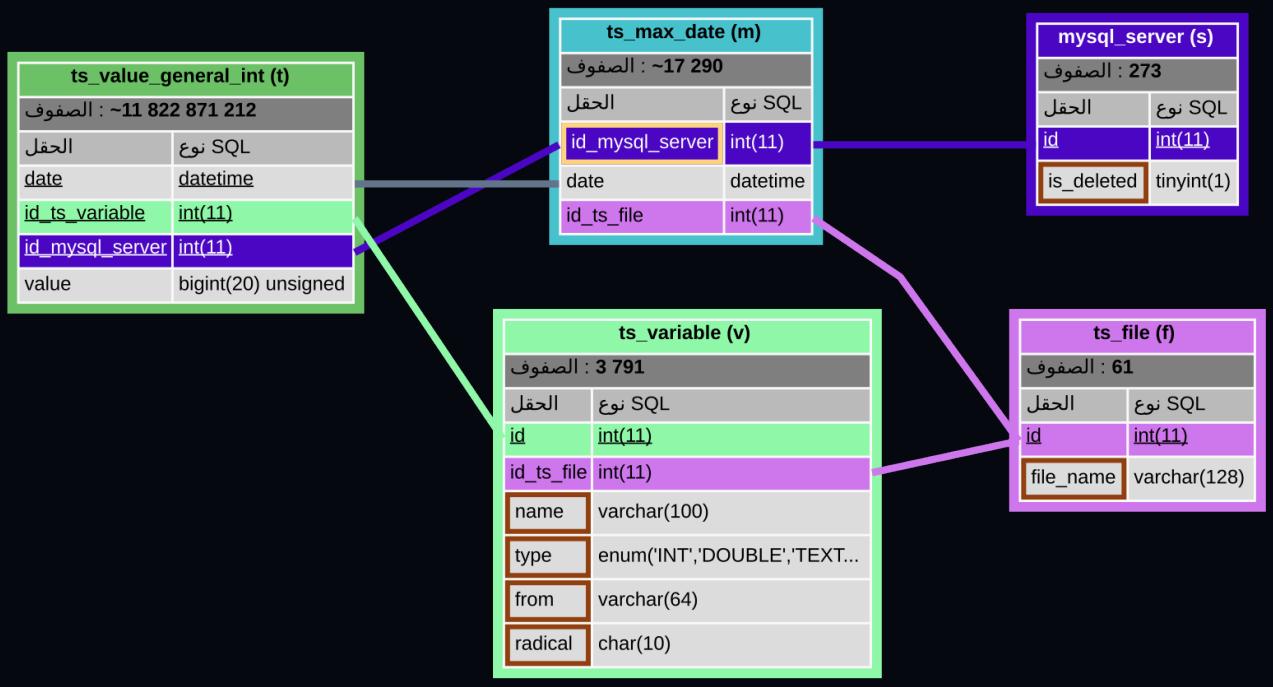
الرسم التراكمي مرجعاً لحجم الاستدعاءات نفسه، ويستخدم كل زوج في الاختبار زمنه الأول.

المؤشر	الساعة السابقة	الساعة التالية
عدد الاستدعاءات التقديري	126,229	123,241
التنفيذات في الدقيقة	2,104	2,054
متوسط الزمن المرجح	18.43 ms	✓ **0.70 ms (÷26.25)**
ثواني SQL التراكمية	2,326.5	✓ **86.5 (÷26.88)**
التنفيذات لكل ثانية SQL	54	✓ **1,424 (×26.25)**
الصفوف المفحوصة لكل استدعاء	2.99	✓ **1.00 (÷2.98)**

ارتفعت الكفاءة، محسوبة بعدد القراءات لكل ثانية من زمن SQL التراكمي، من **54 إلى 1424**. وانخفض متوسط زمن التنفيذ **96.2%** في حين انخفض معدل التنفيذ المرصود **2.4%** فقط. تعبر الكفاءة عن كلفة القراءة، بينما يتأثر المعدل الفعلي أيضاً بتردد الفحص ودرجة التوازي.

ما الذي يبحث عنه الاستعلام؟

يعطي جدول `ts_max_date` آخر وقت لملف القياسات على الخادم المطلوب. ويحدد `ts_file` الملف و `ts_variable` المقياس، بينما يستبعد `mysql_server` الخوادم المحذوفة. ثم يبحث الربط الخارجي عن القيمة الصحيحة المسجلة في ذلك الوقت للمتغير والخادم نفسيهما. وإذا غاب القياس فقد يعيد هذا الربط `NULL`.



مخطط الاستعلام المعدّل كما يعرضه PmaControl. تصل الأسهم بين أعمدة الربط، ويحدد الإطار البني الحقول التي عليها شروط تصفية، ومنها `ts_variable.from`. تشير العلامة `~` إلى أن عدد الصفوف تقديري وقت أخذ المخطط.

تعريفات الجداول الخمسة

هذه نتائج الأمر `SHOW CREATE TABLE` على خادم MariaDB المستخدم في هذه الدراسة. في تعريف `ts_variable` يعرض السطر `UNIQUE KEY` العمودين اللذين يستخدمهما الاستعلام المعدّل بترتيبهما في الفهرس.

`SHOW CREATE TABLE ts_variable;`

```
CREATE TABLE `ts_variable` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_ts_file` int(11) NOT NULL DEFAULT 3,
  `name` varchar(100) NOT NULL,
  `type` enum('INT','DOUBLE','TEXT','JSON','') NOT NULL COMMENT '1 => int, 2 => double, 3 =>
text, 4=> json',
  `from` varchar(64) NOT NULL,
  `radical` char(10) NOT NULL,
  `is_derived` int(11) DEFAULT 1,
  `is_dynamic` int(11) DEFAULT 1 COMMENT 'only for global variables',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`,`from`),
  KEY `ts_variable_ibfk_1` (`id_ts_file`),
```

```
  KEY `from` (`from`),
  CONSTRAINT `ts_variable_ibfk_1` FOREIGN KEY (`id_ts_file`) REFERENCES `ts_file` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=39387 DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```

SHOW CREATE TABLE ts_file;

```
CREATE TABLE `ts_file` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `file_name` varchar(128) NOT NULL,
  `each` int(11) NOT NULL DEFAULT 1,
  PRIMARY KEY (`id`),
  UNIQUE KEY `file_name` (`file_name`)
) ENGINE=InnoDB AUTO_INCREMENT=25508473 DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```

SHOW CREATE TABLE ts_max_date;

```
CREATE TABLE `ts_max_date` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_mysql_server` int(11) NOT NULL,
  `date` datetime DEFAULT NULL,
  `sample_epoch` bigint(20) unsigned NOT NULL DEFAULT 0,
  `date_p1` datetime DEFAULT NULL,
  `sample_epoch_p1` bigint(20) unsigned NOT NULL DEFAULT 0,
  `date_p2` datetime DEFAULT NULL,
  `sample_epoch_p2` bigint(20) unsigned NOT NULL DEFAULT 0,
  `date_p3` datetime DEFAULT NULL,
  `sample_epoch_p3` bigint(20) unsigned NOT NULL DEFAULT 0,
  `date_p4` datetime DEFAULT NULL,
  `sample_epoch_p4` bigint(20) unsigned NOT NULL DEFAULT 0,
  `id_ts_file` int(11) NOT NULL,
  `last_date_listener` datetime DEFAULT current_timestamp(),
  `last_sample_epoch_listener` bigint(20) unsigned NOT NULL DEFAULT 0,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_mysql_server_2` (`id_mysql_server`,`id_ts_file`),
  UNIQUE KEY `id_mysql_server` (`id_mysql_server`,`id_ts_file`,`date`),
  UNIQUE KEY `date` (`date`,`id_ts_file`,`id_mysql_server`),
  KEY `id_ts_file` (`id_ts_file`),
  KEY `id_mysql_server_4` (`id_mysql_server`,`date`),
  KEY `date_2` (`date`,`last_date_listener`),
  CONSTRAINT `ts_max_date_ibfk_3` FOREIGN KEY (`id_ts_file`) REFERENCES `ts_file` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=22859 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

`SHOW CREATE TABLE mysql_server;`

```
CREATE TABLE `mysql_server` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `is_deleted` tinyint(1) NOT NULL DEFAULT 0,
  `created_at` datetime NOT NULL DEFAULT current_timestamp(),
  `updated_at` datetime NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `id_client` int(11) NOT NULL,
  `id_environment` int(11) NOT NULL,
  `name` varchar(100) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci NOT NULL,
  `display_name` varchar(100) NOT NULL,
  `ip` varchar(255) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL,
  `hostname` varchar(200) NOT NULL DEFAULT '',
  `login` varchar(32) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci NOT NULL,
  `passwd` varchar(255) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL,
  `database` varchar(64) NOT NULL,
  `is_password_crypted` int(11) NOT NULL,
  `is_ssl` int(11) NOT NULL DEFAULT 0,
  `port` int(11) NOT NULL,
  `ssh_nat` varchar(64) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL DEFAULT '',
  `ssh_port` int(11) NOT NULL DEFAULT 22,
  `ssh_login` text CHARACTER SET latin1 COLLATE latin1_bin NOT NULL DEFAULT '\'\'',
  `is_sudo` int(11) NOT NULL DEFAULT 0,
  `is_root` int(11) NOT NULL DEFAULT 1,
  `is_monitored` int(11) NOT NULL DEFAULT 1,
  `monitoring_disabled_reason` enum('manual','license_quota') DEFAULT NULL,
  `is_proxy` int(11) NOT NULL DEFAULT 0,
  `is_vip` tinyint(1) NOT NULL DEFAULT 0,
  `is_binlog_relay` tinyint(1) NOT NULL DEFAULT 0,
  `is_clickhouse` tinyint(1) NOT NULL DEFAULT 0,
  `errant_gtid_set` text DEFAULT NULL COMMENT '#1280 P0.1 last computed errant GTID set
(canonical form, empty string = no errant)',
  `is_acknowledged` int(11) NOT NULL DEFAULT 0,
  `timeout` int(11) GENERATED ALWAYS AS (case when `is_proxy` = 1 then 11 else 1 end)
VIRTUAL,
  `errant_gtid_sampled_at` datetime DEFAULT NULL COMMENT '#1280 P0.1 wall-clock of the last
successful errant-set computation',
  `errant_gtid_ignore` tinyint(1) NOT NULL DEFAULT 0,
  `replica_lag_sla_seconds` int(11) NOT NULL DEFAULT 30,
  `is_out_of_service` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'operator-flagged HS for reload
from master',
  PRIMARY KEY (`id`),
```

```
  UNIQUE KEY `name` (`name`),
  UNIQUE KEY `ip` (`ip`,`port`),
  KEY `id_client` (`id_client`),
  KEY `id_environment` (`id_environment`),
  KEY `is_monitored` (`id_client`,`is_monitored`,`is_deleted`) USING BTREE,
  KEY `is_deleted` (`is_deleted`)
) ENGINE=InnoDB AUTO_INCREMENT=337 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci WITH
SYSTEM VERSIONING
 PARTITION BY SYSTEM_TIME
PARTITIONS 2;
```

SHOW CREATE TABLE ts_value_general_int;

```
CREATE TABLE `ts_value_general_int` (
  `date` datetime NOT NULL,
  `id_ts_variable` int(11) NOT NULL,
  `id_mysql_server` int(11) NOT NULL,
  `value` bigint(20) unsigned DEFAULT NULL,
  PRIMARY KEY (`date`,`id_ts_variable`,`id_mysql_server`),
  KEY `id_mysql_server` (`id_mysql_server`,`id_ts_variable`,`date`)
) ENGINE=ROCKSDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
 PARTITION BY RANGE (to_days(`date`))
(PARTITION `p740279` VALUES LESS THAN (740279) ENGINE = ROCKSDB,
 PARTITION `p740280` VALUES LESS THAN (740280) ENGINE = ROCKSDB,
 PARTITION `p740281` VALUES LESS THAN (740281) ENGINE = ROCKSDB,
 PARTITION `p740282` VALUES LESS THAN (740282) ENGINE = ROCKSDB,
 PARTITION `p740283` VALUES LESS THAN (740283) ENGINE = ROCKSDB,
 PARTITION `p740284` VALUES LESS THAN (740284) ENGINE = ROCKSDB,
 PARTITION `p740285` VALUES LESS THAN (740285) ENGINE = ROCKSDB,
 PARTITION `p740286` VALUES LESS THAN (740286) ENGINE = ROCKSDB,
 PARTITION `p740287` VALUES LESS THAN (740287) ENGINE = ROCKSDB,
 PARTITION `p740288` VALUES LESS THAN (740288) ENGINE = ROCKSDB,
 PARTITION `p740289` VALUES LESS THAN (740289) ENGINE = ROCKSDB,
 PARTITION `p740290` VALUES LESS THAN (740290) ENGINE = ROCKSDB,
 PARTITION `p740291` VALUES LESS THAN (740291) ENGINE = ROCKSDB,
 PARTITION `p740292` VALUES LESS THAN (740292) ENGINE = ROCKSDB,
 PARTITION `p740293` VALUES LESS THAN (740293) ENGINE = ROCKSDB,
 PARTITION `p740294` VALUES LESS THAN (740294) ENGINE = ROCKSDB,
 PARTITION `p740295` VALUES LESS THAN (740295) ENGINE = ROCKSDB,
 PARTITION `p740296` VALUES LESS THAN (740296) ENGINE = ROCKSDB,
 PARTITION `p740297` VALUES LESS THAN (740297) ENGINE = ROCKSDB,
```

```
 PARTITION `p740298` VALUES LESS THAN (740298) ENGINE = ROCKSDB,
 PARTITION `p740299` VALUES LESS THAN (740299) ENGINE = ROCKSDB,
 PARTITION `p740300` VALUES LESS THAN (740300) ENGINE = ROCKSDB,
 PARTITION `p740301` VALUES LESS THAN (740301) ENGINE = ROCKSDB,
 PARTITION `p740302` VALUES LESS THAN (740302) ENGINE = ROCKSDB,
 PARTITION `p740303` VALUES LESS THAN (740303) ENGINE = ROCKSDB,
 PARTITION `p740304` VALUES LESS THAN (740304) ENGINE = ROCKSDB,
 PARTITION `p740305` VALUES LESS THAN (740305) ENGINE = ROCKSDB,
 PARTITION `p740306` VALUES LESS THAN (740306) ENGINE = ROCKSDB,
 PARTITION `p740307` VALUES LESS THAN (740307) ENGINE = ROCKSDB,
 PARTITION `p740308` VALUES LESS THAN (740308) ENGINE = ROCKSDB,
 PARTITION `p740309` VALUES LESS THAN (740309) ENGINE = ROCKSDB,
 PARTITION `p740310` VALUES LESS THAN (740310) ENGINE = ROCKSDB);
```

التغيير في المجمّع

يحمل فهرس جدول `ts_variable` الفريد اسم `name` ويضم **(`name`, `from`)**. يمكن لمصادر مختلفة استخدام اسم المقياس نفسه، بينما يحدد زوج الاسم والمصدر هنا صفاً واحداً على الأكثر. كان المصدر معروفاً عند الكتابة. أما القراءة فكانت تحدد الاسم والنوع والحقل radical وتغفل المصدر. يستخدم الإصدار المنشور المصدر نفسه عند القراءة. وفيما يلي استعلام توافر MySQL مع مُعامِل لمعرف الخادم.

```
SELECT t.value AS previous_value
FROM ts_max_date AS m
JOIN ts_file AS f ON f.id = m.id_ts_file
JOIN ts_variable AS v ON v.id_ts_file = f.id
  AND v.name = 'mysql_available'
  AND v.`from` = 'mysql_server'  /* + */
  AND v.type = 'INT' AND v.radical = 'general'
JOIN mysql_server AS s ON s.id = m.id_mysql_server
  AND s.is_deleted = 0
LEFT JOIN ts_value_general_int AS t
  ON t.id_mysql_server = m.id_mysql_server
 AND t.id_ts_variable = v.id
 AND t.date = m.date
WHERE m.id_mysql_server = ? AND f.file_name = 'mysql_server'
LIMIT 1;
```

السطر المعلّم بتعليق هو الإضافة الوحيدة. و `from` هنا اسم عمود محاط بعلامتي اقتباس مع كوستين، وقيمته في هذا المسار هي `mysql_server`. تستخدم المجمّعات الأخرى مصادرها الخاصة. وكان الفهرس موجوداً قبل النشر.

الـ يميّز المصدر سوى 32 مجموعة إضافية بين 3,791 صفاً

جمّعنا صفوف `ts_variable` البالغ عددها 3,791 ضمن لقطة واحدة للقراءة فقط يوم 21 نوفمبر 2026 الساعة 21:25 بتوقيت باريس. النسبة في الجدول هي عدد المجموعات المميزة مقسوماً على عدد صفوف الجدول.

أعمدة التجميع	عدد المجموعات الفعلي	المجموعات / الصفوف	`SHOW INDEX` قبل التحديث وبعده
(`name`)	3,759	99.16%	3,841 → 3,840
(`name` , `from`)	3,791	100%	3,841 → 3,840
(`from`)	42	1.11%	42 → 42

يمثل السطران الأولان البادئتين المتتاليتين للفهرس الفريد `name`. أما الثالث فيخص الفهرس المستقل على `from`. عند القياس الأول، عرضت MariaDB القيمة **3,841 في كل الموضعين** من الفهرس الفريد. هذه إحصائية تقديرية وقد تتجاوز عدد الصفوف الفعلي. مرجع MariaDB لعدد القيم المميزة.

توضح الأعداد سبب هذا التوقع. يظهر **3,729 اسماً** مرة واحدة، و**28** مرتين، و**اسمان** ثلاث مرات. يضيف التجميع حسب (`name` , `from`) **32 مجموعة فقط** إلى الجدول بأكمله. ترتفع نسبة المجموعات المميزة إلى الصفوف من **99.16% إلى 100%**، أي **0.84 نقطة مئوية**. وحتى هذا الفارق الضئيل يختفي عند البحث عن مقياس التوافر. يطابق الاسم `mysql_available` **صفاً واحداً** وعند تحديد المصدر نحصل على **الصف نفسه**.

```
SELECT rows_per_name, COUNT(*) AS number_of_names
FROM (
    SELECT name, COUNT(*) AS rows_per_name
    FROM ts_variable
    GROUP BY name
) AS names
GROUP BY rows_per_name
ORDER BY rows_per_name;
```

بعد تحديث إحصاءات `ts_variable`

في الساعة 21:30 بتوقيت باريس أعاد الأمر `ANALYZE NO_WRITE_TO_BINLOG TABLE pmacontrol.ts_variable` النتيجة `OK` على الخادم المدروس. تغيّر `SHOW INDEX` قبل التحديث وبعده من 3,841 إلى **3,840 لكلتا البادئتين**. وعند إعادة التجميع مباشرةً بعد التحديث بقي عدد المجموعات الفعلي كما هو.

تخزّن InnoDB تقديرات البادئات في `mysql.innodb_index_stats`. بالنسبة إلى الفهرس `name` تغيّرت قيمة `n_diff_pfx01` من 3,757 إلى **3,759** وقيمة `n_diff_pfx02` من 3,789 إلى **3,791**. تطابق القيمتان الآن العدد الفعلي للمجموعات. قبل التحديث وبعده قدّر `EXPLAIN` صفاً واحداً لكل من عمليتي البحث المنفصلتين في `ts_variable`، وبقي نوع الوصول `ref` للاسم و `const` للزوج.

استعلامات العد والنتائج الأصلية (ZIP)

مقارنة خطتي التنفيذ

كانت MariaDB تستخدم الفهرس `name` بطريقة الوصول `ref` على عمود الأول. ومع وجود شرطي المساواة أصبح المفتاح الفريد محدداً بالكامل وتحول الوصول إلى `v` إلى `const`. بلغ طول الجزء المستخدم من المفتاح 102 ثم 168 بايتاً. وتغير الوصول إلى `t` بالمفتاح الأساسي من `eq_ref` إلى `const`. وانخفض عدد الصفوف المفحوصة في الاستعلام كله من نحو ثلاثة إلى صف واحد لكل استدعاء. يستطيع المحسّن التحقق من التفرد قبل إكمال عمليات الربط.

تعرض كل خلية ترتيب القراءة ونوع الوصول والفهرس المختار وطول المفتاح بالبايت. يعرض كل صف الجدول نفسه في الخطتين.

الجدول	قبل	بعد
`ts_file` (f)	1 · `const` · `file_name` · 130 B	1 · `const` · `file_name` · 130 B
`ts_max_date` (m)	2 · `const` · `id_mysql_server_2` · 8 B	3 · `const` · `id_mysql_server_2` · 8 B
`ts_variable` (v)	3 · `ref` · `name` · 102 B	2 · `const` · `name` · 168 B
`mysql_server` (s)	4 · `ref` · `PRIMARY` · 4 B	5 · `ref` · `PRIMARY` · 4 B
`ts_value_general_int` (t)	5 · `eq_ref` · `PRIMARY` · 13 B	4 · `const` · `PRIMARY` · 13 B

في الخطتين، تقدّر MariaDB صفاً واحداً لكل جدول من الجداول الخمسة وتسرد الأقسام الـ32 نفسها لجدول `ts_value_general_int`.

الخطة الكاملة قبل التغيير (CSV) · الخطة الكاملة بعد التغيير (CSV)

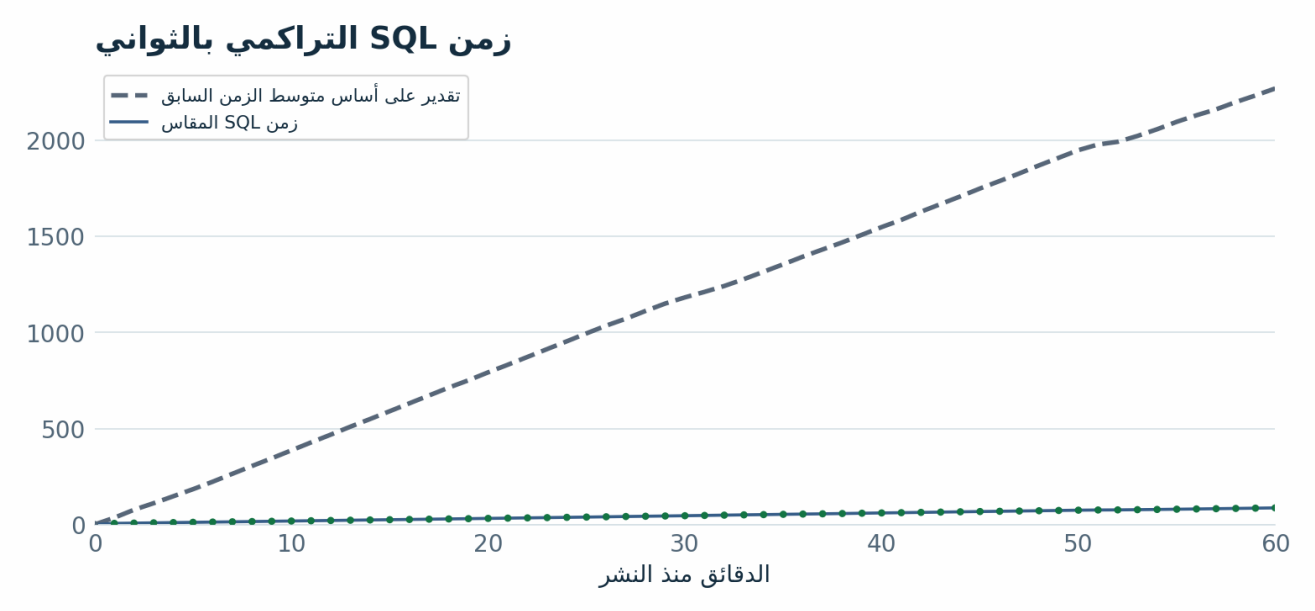
تغير الزمن حسب المرحلة

المرحلة · الوسيط عبر 50 خادماً	قبل	بعد
التحسين	35.085 ms	✓ **0.261 ms (÷134.27)**
التنفيذ بعد التحسين	0.107 ms	0.011 ms
مرحلة `Statistics`	54.983 ms	✓ **0.224 ms (÷245.46)**

السطران الأولان من `ANALYZE FORMAT=JSON`. قيس زمن مرحلة `Statistics` بشكل مستقل باستخدام `SHOW PROFILE`.

عند تحديد `name` وحده **تقدّر** MariaDB أنها ستقرأ صفاً واحداً. وعند تحديد الشرطين **يضمن** الفهرس الفريد (`name`, `from`) مطابقة صف واحد على الأكثر. لذلك يمكن اعتبار `v.id` ثابتاً أثناء إعداد خطة الربط بجدول القيم. وعلى 50 خادماً **انخفض وسيط زمن إعداد الخطة من 35.085 إلى 0.261 ملي ثانية**، وانخفض زمن التنفيذ اللاحق من **0.107 إلى 0.011 ملي ثانية**. تحقق معظم الوفر المقاس أثناء إعداد خطة الربط.

زمن SQL التراكمي منذ النشر

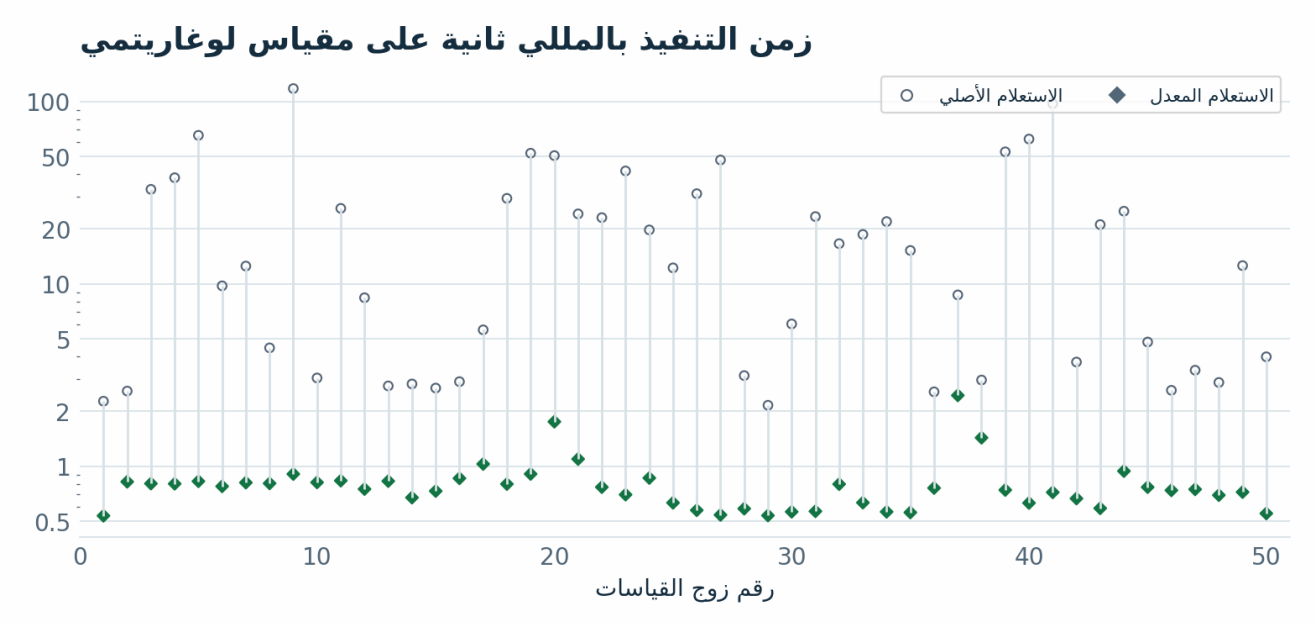


يبدأ المنحنيان من الصفر عند النشر ويستخدمان عدد الاستدعاءات الفعلي نفسه بعده. يجمع الخط المتصل الزمن المقاس، أما المرجع فيطبق متوسط كلفة الساعة السابقة على هذه الاستدعاءات. بعد ساعة بلغ **الزمن المقاس 86.5 ثانية** مقابل **2271.4 ثانية مقدرة** بالكلفة السابقة، أي **وفر تقديري قدره 36 دقيقة و25 ثانية من زمن SQL**. قد تتداخل أزمنة الجلسات ويشمل القياس فترات الانتظار. ولا يعبّر هذا التقدير عن دقائق المعالج.

اختبار مقترن على 50 خادماً

على MariaDB 11.8.10 مع تعطيل ذاكرة التخزين المؤقت للاستعلامات، اخترنا عشوائياً 50 معرفاً مختلفاً من بين 111 معرفاً يعيد قيمة. قرأ استعلام كل زوج الخادم نفسه في معاملة للقراءة فقط. نُفذ الاستعلام الأصلي أولاً في 25 زوجاً، والمعدل أولاً في الأزواج الخمسة والعشرين الأخرى. أخذنا الأزمنة من `SHOW PROFILES` على الخادم. **تطابقت النتائج في الأزواج الخمسين** وكان المعدل **أسرع في كل زوج**.

النسخة	الأدنى (ms)	الوسيط (ms)	المتوسط (ms)	الأعلى (ms)
قبل	2.159	12.543	21.686	117.530
بعد	✓ 0.535 (÷4.03)	✓ 0.755 (÷16.62)	✓ 0.804 (÷26.98)	✓ 2.446 (÷48.05)



يصل كل عمود بين زمني القياس للخادم نفسه. ويُظهر المقياس اللوغاريتمي القيم التي تقل عن ملي ثانية والقيمة القصوى البالغة 117.53 ملي ثانية. بلغت نسبة المتوسطين في العينة 26.98، ونسبة المتوسطين المرجحين في ساعتي التشغيل 26.25.

مقارنة الصيغ بالاستعلام المنشور

يقارن كل صف متوسط زمن الاستعلامين على الخوادم نفسها في التجربة نفسها. تشمل كل سلسلة 50 خادماً. يبين العدد بين قوسين عدد عمليات التنفيذ لكل صيغة.

الصيغة	المرجع (مللي ثانية)	الصيغة (مللي ثانية)
الاسم فقط (150)	0.651	✗ **36.782 (×56.50)**
الاسم فقط مع فرض المفتاح الأساسي لجدول القيم (150)	0.651	✗ **1.674 (×2.57)**
المفتاح الكامل مع فرض المفتاح الأساسي لجدول القيم (150)	0.651	0.630
فرض الفهرس المغطي على جدول التواريخ (50)	0.692	✗ **20.232 (×29.25)**
التحقق من الخادم النشط باستخدام `EXISTS` (150)	0.651	0.663
قراءة القيمة باستعلام فرعي عددي (150)	0.617	0.546
فرض ترتيب الربط (50)	0.692	✗ **41.737 (×60.34)**
استعلام فرعي عددي مع `EXISTS` (150)	0.617	0.563
استعلام فرعي عددي دون `LIMIT` (150)	0.617	0.560
عمليات الربط المنشورة دون `LIMIT` (150)	0.617	0.602
استعلام فرعي عددي بالاسم وحده (150)	0.617	1.218

قياسات الصيغ والمراحل بعد إخفاء هوية الخوادم (ZIP)

كيفية إعادة حساب الرسوم

تمتد النافذة من 16:26:47 إلى 18:26:47 بتوقيت باريس. قُدرت العدادات عند حدود الفترات بالاستيفاء بين اللقطات، وثُبتت نقطة الانتقال بأعداد الاستدعاءات المقروءة مباشرة عند النشر. تشمل النتائج 496 استدعاءً تجريبياً للصمة الجديدة قبل النشر و224 استدعاءً أخيراً للصمة القديمة بعده. وأثرت إعادة تشغيل المجمّعات نحو 18:17 أيضاً في نهاية منحنى عدد الاستدعاءات.

تعطي الزيادة في `COUNT_STAR` خلال دقيقة واحدة عدد الاستدعاءات في تلك الدقيقة مباشرة. ولحساب متوسط زمن التنفيذ نقسم الزيادة في `SUM_TIMER_WAIT` على الزيادة في `COUNT_STAR`

ثم على مليار للتحويل من البيكوثانية إلى المللي ثانية. ويُحسب المتوسط الساعي من الزمن الكلي والعدد الكلي. يعزل الاختبار المقترن أثر تعديل SQL بدقة أكبر مما تتيحه ساعتان من المراقبة أثناء التشغيل.

تنزيل البيانات والخطط

- لقطات العدادات (CSV)
- القياسات الدقيقة بدقيقة والزمن التراكمي (CSV)
- 50 زوجاً مجهول الهوية (CSV)
- الخطط التفصيلية (CSV)
- توثيق EXPLAIN في MariaDB